Lab_11_KMeans.m

```matlab
%% Machine Learning
% Lab 11: K-Means Algorithm
% --- K-Means and Image Compression ---
%{
In this exercise, you will implement the K-means clustering algorithm and
apply it to compress an image.
%}

%% Initialization
clear ; close all; clc

%% ================= Part 1: Find Closest Centroids ====================
%  To help you implement K-Means, we have divided the learning algorithm
%  into two functions -- findClosestCentroids and computeCentroids. In this
%  part, you should complete the code in the findClosestCentroids function.
%
fprintf('Finding closest centroids.\n\n');

% Load an example dataset that we will be using
load('ex7data2.mat');

% Select an initial set of centroids
K = 3; % 3 Centroids
initial_centroids = [3 3; 6 2; 8 5];

% Find the closest centroids for the examples using the
% initial_centroids
idx = findClosestCentroids(X, initial_centroids);

fprintf('Closest centroids for the first 3 examples: \n')
fprintf(' %d', idx(1:3));
fprintf('\n(the closest centroids should be 1, 3, 2 respectively)\n');

fprintf('Program paused. Press enter to continue.\n');
pause;

%% ===================== Part 2: Compute Means =========================
%  After implementing the closest centroids function, you should now
%  complete the computeCentroids function.
%
fprintf('\nComputing centroids means.\n\n');

%  Compute means based on the closest centroids found in the previous part.
centroids = computeCentroids(X, idx, K);

fprintf('Centroids computed after initial finding of closest centroids: \n')
fprintf(' %f %f \n' , centroids');
fprintf('\n(the centroids should be\n');
fprintf('   [ 2.428301 3.157924 ]\n');
fprintf('   [ 5.813503 2.633656 ]\n');
fprintf('   [ 7.119387 3.616684 ]\n\n');

fprintf('Program paused. Press enter to continue.\n');
pause;
```

```matlab
%% =================== Part 3: K-Means Clustering ======================
%  After you have completed the two functions computeCentroids and
%  findClosestCentroids, you have all the necessary pieces to run the
%  kMeans algorithm. In this part, you will run the K-Means algorithm on
%  the example dataset we have provided.
%
fprintf('\nRunning K-Means clustering on example dataset.\n\n');

% Load an example dataset
load('ex7data2.mat');

% Settings for running K-Means
K = 5;
max_iters = 5;

% For consistency, here we set centroids to specific values
% but in practice you want to generate them automatically, such as by
% settings them to be random examples (as can be seen in
% kMeansInitCentroids).
initial_centroids = [3 3; 6 2; 8 5];

% Run K-Means algorithm. The 'true' at the end tells our function to plot
% the progress of K-Means
[centroids, idx] = runkMeans(X, initial_centroids, max_iters, true);
fprintf('\nK-Means Done.\n\n');

fprintf('Program paused. Press enter to continue.\n');
pause;

%% ============= Part 4: K-Means Clustering on Pixels ===============
%  In this exercise, you will use K-Means to compress an image. To do this,
%  you will first run K-Means on the colors of the pixels in the image and
%  then you will map each pixel onto its closest centroid.
%
%  You should now complete the code in kMeansInitCentroids.m
%

fprintf('\nRunning K-Means clustering on pixels from an image.\n\n');

%  Load an image of a bird
A = double(imread('bird_small.png'));

% If imread does not work for you, you can try instead
%   load ('bird_small.mat');

A = A / 255; % Divide by 255 so that all values are in the range 0 - 1

% Size of the image
img_size = size(A);

% Reshape the image into an Nx3 matrix where N = number of pixels.
% Each row will contain the Red, Green and Blue pixel values
% This gives us our dataset matrix X that we will use K-Means on.
X = reshape(A, img_size(1) * img_size(2), 3);

% Run your K-Means algorithm on this data
% You should try different values of K and max_iters here
```

```matlab
K = 2;
max_iters = 5;

% When using K-Means, it is important the initialize the centroids
% randomly.
% You should complete the code in kMeansInitCentroids.m before proceeding
initial_centroids = kMeansInitCentroids(X, K);

% Run K-Means
[centroids, idx] = runkMeans(X, initial_centroids, max_iters);

fprintf('Program paused. Press enter to continue.\n');
pause;


%% ================= Part 5: Image Compression ======================
%  In this part of the exercise, you will use the clusters of K-Means to
%  compress an image. To do this, we first find the closest clusters for
%  each example. After that, we

fprintf('\nApplying K-Means to compress an image.\n\n');


% Find closest cluster members
idx = findClosestCentroids(X, centroids);

% Essentially, now we have represented the image X as in terms of the
% indices in idx.

% We can now recover the image from the indices (idx) by mapping each pixel
% (specified by its index in idx) to the centroid value
X_recovered = centroids(idx,:);

% Reshape the recovered image into proper dimensions
X_recovered = reshape(X_recovered, img_size(1), img_size(2), 3);

% Display the original image
subplot(1, 2, 1);
imagesc(A);
title('Original');

% Display compressed image side by side
subplot(1, 2, 2);
imagesc(X_recovered)
title(sprintf('Compressed, with %d colors.', K));


fprintf('Program paused. Press enter to continue.\n');
pause;
```

findClosestCentroids.m

```matlab
function idx = findClosestCentroids(X, centroids)
%FINDCLOSESTCENTROIDS computes the centroid memberships for every example
%   idx = FINDCLOSESTCENTROIDS (X, centroids) returns the closest centroids
%   in idx for a dataset X where each row is a single example. idx = m x 1
%   vector of centroid assignments (i.e. each entry in range [1..K])

K = size(centroids, 1);
idx=zeros(size(X,1),1);

for i = 1:size(X,1)
    distance = Inf;
    for k = 1:K
        temp2= X(i,:)-centroids(k,:);
        if (temp2*temp2') < distance
            distance = (temp2*temp2');
            temp1 = k;
        end
    end
    idx(i)=temp1;
end
```

computeCentroids.m

```matlab
function centroids = computeCentroids(X, idx, K)
%COMPUTECENTROIDS returns the new centroids by computing the means of the
%data points assigned to each centroid.

[m n] = size(X);

centroids =  zeros(K, n);

for k=1:K
    centroid = zeros(1,n);
    Ck = 0;
    for i=1:m
        if idx(i)==k
            centroid=centroid + X(i,:);
            Ck = Ck+1;
        end
    end
    centroids(k,:)=centroid.*(1/Ck);
end
```

kMeansInitCentroids.m

```
function centroids = kMeansInitCentroids(X, K)
%KMEANSINITCENTROIDS This function initializes K centroids that are to be
%used in K-Means on the dataset X

centroids = zeros(K, size(X, 2));

randidx=randperm(size(X,1));

centroids=X(randidx(1:K),:);
end
```